University of Saskatchewan
Department of Computer Science

Cmpt 330
C Programming Quiz

October 6, 2003

Time: 50 minutes
Total Marks: 53

Professor: A. J. Kusalik
Closed Book

Name: _____

Student Number: _____

Directions:

Answer each of the following questions in the space provided in this exam booklet. If you must continue an answer (e.g. in the extra space on the last page, or on the back side of a page), make sure you clearly indicate that you have done so and where to find the continuation.

Make all written answers legible; no marks can be given for answers which cannot be decrypted. Where a discourse or discussion is called for, be concise and precise.

Use of calculators is not allowed during the exam. Fortunately, you should not need a calculator for completing any of the questions. The last page of the exam contains supplemental information which may be of use in answering some of the questions.

The context for questions is the C programming language as presented in the "C Short Course" given as part of Cmpt330. If you find it necessary to make any assumptions to answer a question, state the assumption with your answer. Note that answers involving constructs in C++, but not C, will not be considered correct.

Marks for each major question are given at the beginning of that question. There are a total of 53 marks (approximately one mark per minute).

Good luck.

For marking use only:

A. _____/6	D. _____/8	G. _____/10
B. _____/10	E. _____/6	
C. _____/4	F. _____/9	
		Total: _____/53

A. (6 marks)

Consider each of the following bitwise operators. Match each operator on the left with the correct explanation on the right. Use a straight line to indicate the match. I.e. draw a straight line between each operator and its corresponding description.

sequence		represents	
&	•	•	XOR (exclusive-OR)
|	•	•	complement
^	•	•	AND
~	•	•	shift right
<<	•	•	shift left
>>	•	•	OR

B. (1+1+4+2+2 = 10 marks)

For each of the following questions give a very short, precise answer.

1. What C statement, other than `goto`, allows you to terminate a loop prematurely ?

2. What is the subscript (index) of the first entry of (first element in) an array?

3. C data types have different sizes (in units of bytes). Some are fixed, and some are dependent on the underlying machine architecture. Consider the following data types:

`int`	`long`
`int *`	`unsigned int`
`char`	`signed char`
`short`	`char * *`

Of the above data types, name two which are fixed (not dependent on machine architecture) in size.

Of the above data types, name two which are dependent in size on machine architecture.

4. Consider the following C program fragment.

```
int x, *i, **n;
char *argv[], c[]="Username";
```

What are the names of all the variables in the above code sample which are pointer variables?

5. What is the scope of an *auto* variable in C?

C. (2+2 = 4 marks)

Consider each of the following C language statements:

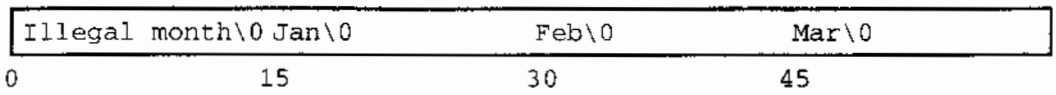
a) `char name[][15] = { "Illegal month", "Jan", "Feb", "Mar" };`

b) `char *name[] = { "Illegal month", "Jan", "Feb", "Mar" };`

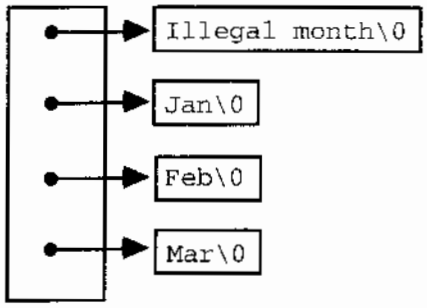
c) `char name[] = { "Illegal month Jan  Feb  Mar" };`

Now consider each of the following diagrams indicating a possible memory layout. For each diagram, indicate which of the above declarations in C will result in that memory layout. Note that one of the above statements will not be used as an answer.

1.



2.



D. (8 marks)

The following program is intended to take the names of two files as its arguments. The first file is opened for reading and the second for writing. The content of the first file is then copied to the second, with the line spacing doubled. Part of the code is missing (i.e. the "blanks" below). Complete the code by filling in the blanks.

```
#include <stdio.h>
#include <stdlib.h>
```

```
void double_space(FILE *, FILE *);
void prn_info(char *);

int main(int argc, char *argv[])
{
   FILE   *ifp, *ofp;

   if (argc ______) {
      prn_info(argv[0]);
      exit(1);
   }

   ifp = fopen(_______, _____);      /* open in file for reading */

   if (ifp _____ NULL) {
      printf ("File error.\n");
      exit (-1);
   }

   ofp = fopen(_______, _____);      /* open out file for writing */

   if (ofp _____ NULL) {
      printf ("File error.\n");
      exit (-1);
   }

   double_space(ifp, ofp);

   if( (fclose(ifp) ______ ) || (fclose(ofp) _____ ) ){
      printf ("File error.\n");
      exit (-1);
   }

   return 0;
}

void double_space(FILE *ifp, FILE *ofp)
{
   int  c;

   while ((c = getc(ifp)) != EOF) {
      putc(c, ofp);
      if (c == _____)
         putc(_____, ofp);
   }

}

void prn_info(char *pgm_name)
{

   printf("\n%s%s%s\n\n%s%s\n\n",
      "Usage:  ", pgm_name, "  infile  outfile",
      "The contents of infile will be double-spaced ",
      "and written to outfile.");
}
```

E. (3+3 = 6 marks)

Knowledge about C programming concepts is necessary to prevent errors when programming. For each of the following C code segments, indicate and describe the error being made. I.e. in each of the following pieces of code, a programming error is being made. Identify (in some easily discernible way) the error in each case. Also describe (by way of a short description) the nature of the error. Then indicate how the error

can be concisely corrected without changing the intended logic of the program sample. Marks will be deducted for reporting non-errors.

1.

```
#include <stdio.h>
#include <string.h>

#define ALPHABET "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

main()
{
   char alpha[27];
   int i, count;

   strcpy( alpha, ALPHABET );
   count = strlen( alpha );
   for( i=1; i <= count; i = i + 1 )
   {
      putchar( alpha[i-1] - 'A' + 'a' );
      putchar( '\n' );
   }
   putchar( '\n' );

   exit( 0 );
}
```

What is the error? How would you most concisely correct it?

2.

```
char *t1="X11R6";
char path[80];

path="/usr/";
printf( "%s\n", t1 );
strcat( path, t1 );
printf( "%s\n", path );
```

What is the error? How would you most concisely correct it?

F. (3+6 = 9 marks)

For each of the following specifications, give a small portion of C code which satisfies it.

1. Given a variable and pointer declared as

```
unsigned int ref_count, *ptr;
```

show with two C language statements how the pointer variable can be used to set the value of the non-pointer variable to 416.

2. Given the following structure type definition, declare a variable, `observation_list`, which is a pointer to such a structure, allocate dynamic memory for 30 records of this type, and have `observation_list` ultimately point to the resultant memory space. Finally, check that the allocation of memory was successful. if it wasn't, print an error message on the appropriate stream.

```
struct observation {
   int id_number;
   char description[20];
   double significance;
};
```

G. (5+3+2 = 10 marks)

Answer each of the following questions with a concise answer.

1. Consider the `char` datatype in C. What is the difference between `""` (empty string) and `(char *)0`? Illustrate with realistic examples. Be precise. Use a diagram.

2. The following statement produces a compiler error:

```
printf( "%d\n", --5 );
```

The error message is

```
invalid lvalue in decrement
```

Explain why there is error; i.e. explain the nature of the error.

3. Suppose that a programmer has written a program to perform some useful operation. When compiled and linked, the executable program will be stored in a file named *xex*. The source code of function `main()` of the program is stored in a file *xex.c*. Various auxilliary functions are stored in file *xex_routines.c*. Finally, definitions of symbols used in *xex.c* and *xex_routines.c* are stored in *xex.h*.

The programmer has also written a *makefile* from creating *xex* from its constituent source code files. Give a line or statement that might appear in this makefile which illustrates a dependency.

Extra Space

(The space below is for answering previous questions or for rough work.)

Supplementary information

You may find the following function prototypes useful in answering some of the questions in this exam:

```
void * calloc(size_t number, size_t size);
int fclose( FILE *stream );
int fgetc( FILE *stream );
char *fgets( char *s, int size, FILE *stream );
FILE *fopen( const char *path, const char *mode );
int fprintf( FILE *stream, const char *format, ... );
int fputc( int c, FILE *stream );
int fputs( const char *str, FILE *stream );
void free( void *ptr );
int fscanf( FILE *stream, const char *format, ... );
int getc( FILE *stream );
int getchar();
char *gets( char *str );
void *malloc( size_t size );
int open( const char *path, int flags, mode_t mode);
void perror( const char *string );
int printf( const char *format, ... );
int putc( int c, FILE *stream );
int putchar( int c );
int puts( const char *str );
void *realloc( void *ptr, size_t size );
int scanf( const char *format, ... );
int sprintf( char *str, const char *format, ...);
char * strcat( char *target, const char *append );
```

Comic Relief